YTÜD 2002/4

ARAŞTIRMA MAKALESİ

SUPERVISOR DESIGN, WHICH IS BASED ON PETRINETS FOR SEQUENTIAL CONTROL OF THE MANUFACTURING SYSTEM BY USING PROGRAMMABLE LOGIC CONTROLLER

Galip CANSEVER*, Y. M. Li, İbrahim Beklan KÜÇÜKDEMİRAL*, A. H. Jones****

*****_Yıldız Teknik Üniversitesi, Elektrik-Elektronik Fakültesi, Elektrik Mühendisliği Bölümü, Beşiktaş – İSTANBUL_
*The Salford University,Department of Aeronautical Eng.(UK)

Geliş Tarihi: 03.08.2001

PROGRAMLANABİLİR LOJİK KONTROLÖRLER İÇİN PETRİ NET TABANLI DENETLEYİCİ TASARIMI

ÖZET

Bu makale ayrık zamanlı olaylar ve onların merdiven tipi lojik diyagramları için sistematik bir denetleyici tasarımı sunmaktadır. Metot, sistem modellemede kullanılan Petri Netlere dayanmaktadır. Metot kontrol yönteminin belirlenmesini, yöntemin sadeleştirilmesini içermektedir. Orijinal modele kavislerin eklenmesi ve serbest bırakılmasını da kapsayan derlenmiş kontrolörün oluşturulmasında Espresso yazılımı kullanılmaktadır. Daha sonra bu metot merdiven tipi lojik kodun sürülmesinde kullanılmıştır. Tüm yöntemi açıklamak için tasarım örnekleri sunulmuştur.

SUMMARY

This paper proposes a systematic approach for design of supervisory controller for Discrete Event Systems and their Ladder Logic Diagrams. The method is based on Petri Nets, which is used to model the systems. It involves defining the control policy, simplifying the control policy using Espresso software to form the compiled controller by adding inhibiting and enabling arcs to the original model. Then the method is used to drive structured Ladder Logic code. A design example is presented to clarify the full procedure.

1. INTRODUCTION

Ramadge and Wonham have studied supervisory control theory for more than 10 years [1,2]. The first approach worked with finite state machine (FSM) where all of the concepts associated with the design of supervisors for discrete event systems were fully described.

In general, a design and implementation method for DES controller must provide tools for the system specification, modeling, analysis and implementation. We use Petri nets as our basic modeling tool because Petri nets have several advantages over FSM [3]. First of all, the states of a Petri net are represented by the possible markings but not by the places, thus they give a compact description, i.e., the structure of the net may be maintained small even if the number of the markings grow. Secondly, instead of using ambiguous textual descriptions or mathematical notations, which can be difficult to understand, the DES and the specifications can be represented graphically using Petri

nets. Finally, by using Petri net models, the same model can be used for the analysis of behavioral properties and performance evaluation as well as for systematic construction of the discrete event controller.
Our method involves creating the reachability graph, specifying the required behavior, obtaining the control policy and applying Espresso logic reduction is used to obtain the control policy in the form of steering places, which are then added to original model in the form of enabling and inhibiting arcs, to create the supervisor.

2. THE DESIGN METHOD

One approach to the design of a supervisor is to formulate the control policy such that it is added to the original Petri net to form the supervisor. Such controllers have been defined by Moody et al [4] using place invariant. However, such an approach adds places to the original Petri net and is not always maximally permissive. The design method presented in this paper also makes use of the original Petri net to form the controller, however, in this case only enabling arcs and inhibiting arcs are added to original Petri net to form the supervisor and the supervisor can be guaranteed to be maximally permissive in all cases.
The approach to design supervisory controller for a discrete event system such that the system does not enter any forbidden markings and is maximally permissive is to take the initial marking of the discrete event system and generate the next set of possible markings from the model by firing either controllable or uncontrollable transitions one at a time (sequential solution). If a bad marking is found during this process, the bad marking is blocked from further expansion. This process generates the blocked reachability graph (BRG) of the discrete event system. Once this has been done the blocked reachability graph can be scanned for forbidden markings. Each forbidden marking can then be traced back through the reachability graph until a controllable transition is found. This controllable transition can then be used to block the forbidden marking and as such formulates one part of the control policy. The complete control policy can be obtained by back tracking all of the forbidden markings to controllable transitions. The control policy can then be refined by considering each controllable transition in turn and collecting together all of the markings which require the transition to be blocked (good to bad) and all of the markings which require the transition to be enabled (good to good).
By collecting together two sets namely the set to be blocked and the set to be enabled it is possible to simplify the control policy for each transition by using logic reduction. Logic reduction techniques sub divide the problem into three distinct types namely blocking, enabling and don't care. One of the best-known logic reduction techniques was developed by Brayton et al [5]. The technique provides an optimal minimal logic solution to the logic reduction problem. The solution is in the form of logical expressions involving AND, OR and NOT. Once these logical expressions have been obtained they represent minimal logic solution of the control policy for the controllable transition. In order to translate the results of the Espresso algorithm into a format suitable for generating a Petri net supervisor the Petri net features of enabling and inhibiting arcs are used.
The control policy obtained from Espresso is in the form of steering places, which are involved predicate logic description. These places contain two types of places, i.e., "enabling places" and "inhibiting places". An "enabling place" implies its presence enables the transition. An "inhibiting place" implies its absence enables the transition. These two types of place individually influence each controllable transition. The control policy is thus a unique set of steering places, which will *enable* a particular controllable transition.

To enable a controllable transition from one of the corresponding “enabling places”, an enabling arc is used; to enable a controllable transition from one “inhibiting places”, an inhibiting arc is used.
The advantage of implementing the control policy using enabling and inhibiting arcs from the steering places is that this results in the controllable transition being enabled from the steering places in accordance with the control policy, and it ensures that the tokens at the steering places are unaffected by the control policy when the transition is fired.
This features a very important as it ensures that the addition of the control policy in the form of enabling and inhibiting arcs ensures that the control policy is implemented in a manner such that the forbidden states are blocked and the maximally permissive behavior is not compromised. As such they represent the perfect implementation strategy for the control policy, which emerges from the Espresso algorithm.
Once each controllable transition is processed to obtain the control policy via the Espresso algorithm and the results of Espresso are implemented as enabling and inhibiting arcs [6], the Petri net supervisor naturally emerges in a complied Petri net form. The Petri net supervisor is thus the original Petri net on to which additional enabling and inhibitor arcs have been added. The blocked reachability graph of the Petri net supervisor will exhibit maximally permissive behavior and will not enter any forbidden markings. In addition the supervised reachability graph can also be used to verify the Petri net supervisor.
As a summary of the method stated above, the design technique involves the following five steps:

- Design the uncontrolled model of the system.
- Generate the Blocked Reachability Graph of the model.
- Search the BRG to define the control policy for each controllable transition.
- Apply Espresso minimization technique to the control policy
- Generate the supervisor by adding enabling and inhibiting arcs, to the original Petri net model.

3. LOGIC MINIMIZATION OF CONTROL POLICY

Once the control policy has been formulated for controllable transition in terms of markings either enabling or inhibiting the transitions. It is desirable to reduce the control policy to its simplest form, as this will provide the most transparent and efficient solution. The control policy can be implemented using enabling and inhibiting arcs to either enable or block controllable transitions.
In most cases it is possible to use the rules of logic reduction to simplify the control policy to its simplest form. The result is an expression with the fewest literals and thus has the simplest implementation in terms of a minimum number of enabling and inhibiting arcs.
Because the Espresso method extracts a set of steering places which covers the on-set, this on-set is represented by the set of markings which do not lead to forbidden markings, and the control policy essentially enforces an enabling strategy. However, by default this policy also blocks markings, which lead to forbidden states. This is because the process is reliant on collecting together two sets of markings for each controllable transition, namely a set of markings which lead to forbidden markings defined as $M_{b,\,ti}$, where $ti \in \forall$ controllable transitions, and a set of markings which do not lead to forbidden markings defined as $M_{e,\,ti}$.

Step 1. Design the uncontrolled model of the system using Petri nets

Petri net model of Fig. 1 is for the example of partially controllable net. It is consist of 7 places and 6 transitions. The initial marking is:

$M_1 = (2\ 2\ 0\ 0\ 0\ 1\ 0)^T$

It is assumed that controllable transitions are to be t_1, t_2 and t_5, and the uncontrollable transitions are to be t_3, t_4 and t_6. The forbidden state is: $M(p_5) + M(p_7)<=1$, that means both places p_5 and p_7 can not contain token(s) simultaneously.

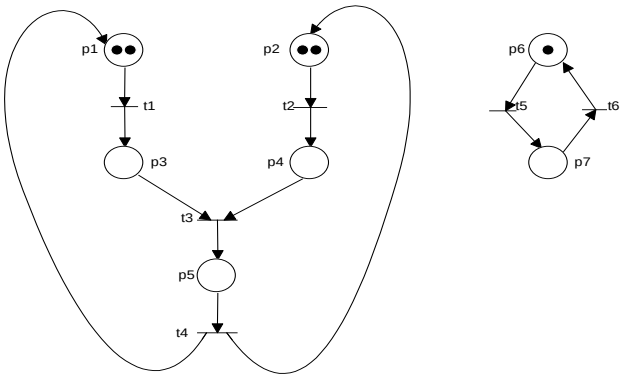


Figure 1. PN model of Partially controllable system

Step 2. Generate the blocked reachability graph of the model

The Blocked Reachability Graph of the Petri net model, in which each node represents a marking reachable from the initial marking M_1 through the sequential transitions and each arc represents the firing of transition, is generated in Fig. 2.
The BRG has been developed by checking if the current marking is a forbidden state. If the marking is a forbidden state it is blocked and no further expansion from that marking takes place.

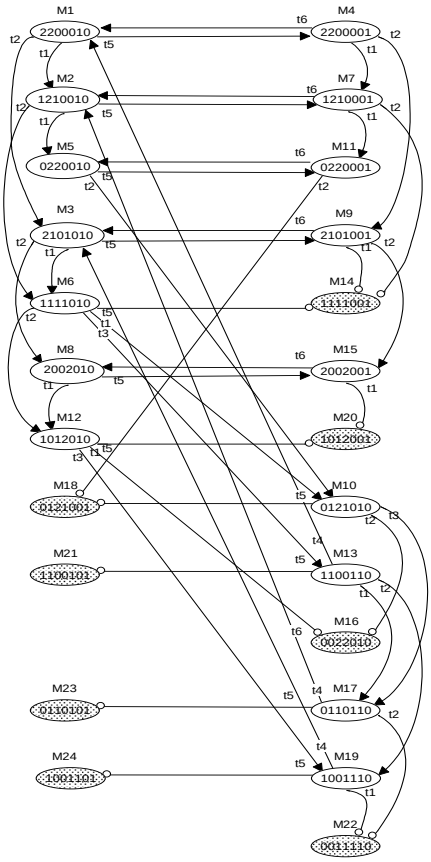


Figure 2. The Blocked Reachability Graph of the Model

In the partially controllable Petri net not only the actual forbidden states should be avoided but also all other states from which a forbidden state can be uncontrollably reached must be avoided. These states are called Co-forbidden states.

Step 3. Search the BRG to define the control policy for each controllable transition

For each controllable transition, the markings are configured in the form of either good-to-good (arrow arcs of Fig.2) or good-to-bad (circle arcs of Fig.1). Good-to-good represents markings of required behavior and good-to-bad represents markings from the required behavior to a forbidden state.

Step 4. Apply the Espresso minimization technique to each controllable transition within the control policy

For each transition, each of the good-to-good and good-to-bad markings is passed to the Espresso algorithm. In this case the controllable transition t_1, t_2 and t_5 result in any enabling or blocking behavior.

Step 5. Generate the supervisory controller by adding enabling and inhibiting arcs to the Petri nets model.

The Espresso method searches a set of steering places for each controllable transition. In this case there are 3 controllable transitions t_1, t_2 and t_5. The control policy is represented by adding two extra controllable transitions for both t_1 and t_2, and one extra transition for t_5. Using the steering places results from the Espresso algorithm, the influence of each steering place is added to the original Petri net by using either an enabling or a inhibiting arc to the appropriate controllable transition. Fig.3 shows the supervisory controlled Petri net. This represents the first maximally permissive compiled supervisory solution to this problem.

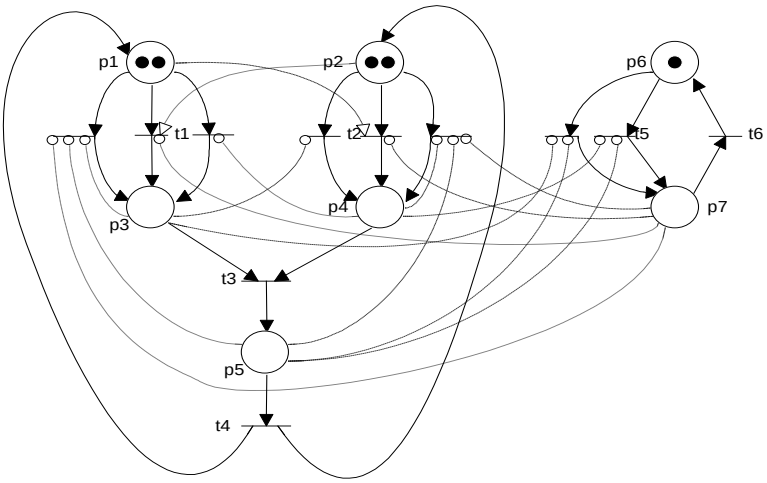


Figure 3. The controlled Petri net model (supervisor) of Figure 1.

4. CONVERSION OF THE MULTI - LEVEL LOGIC THE PETRI NET MODEL TO IEC 1131-3 LADDER DIAGRAMS IN SUPERVISORY CONTROL OF DISCRETE EVENT SYSTEMS

When developing complex control systems involving multi products and parallel tasks, which interact periodically, programmable logic controllers have emerged as the mainstay in the execution of automation tasks. Therefore, petri nets have appeared as the most promising tool to facilitate such design work. In order to facilitate the conversion of on petri net into an LD program, conforming to IEC 1131-3 the token passing logic(TPL) concept can be used. IEC 1131-3 is the only global standard for industrial control programming. It harmonizes the way people design and operate industrial controls by standardizing the user interface. The standard include the definition of Sequential Function Chart (SFC), used to structure the internal organization of a program, and four inter-operable languages: Instruction List (IL), Ladder Diagram (LD) Function Block Diagram (FBD) and Structured Text (ST).
The implementation is carried out by using an ALLEN-BRADLEY MicroLogix 1000 PLC. The Token Passing Logic given in Fig.4 for the “Petri Net Model of Supervisory Controller” is obtained by applying the TPL concept to the APN model. In this case, transition firings are realized by inputs I0/0, I0/1, I0/2…. And also output bits, O:0/0,

O:0/1, O:0/2…. , up and down counter are assigned to the places for the output instructions that let control operations number of passing tokens. They also include preset and accumulator values for up counting to places and down counting from places. At the same time, compare instructions are realised which is being input conditions on the transitions, EQU, NEQ, LES, LEQ, GRT, GEQ, MEQ, and LIM.
The most successful solutions to the problem have involved the use of petri net for the conceptual design. Because of the success of petri net designs there have been some attempts to produce structured methods to convert petri nets into ladder logic. However until the advent of the token passing ladder logic no solution has been provided to cope with the full complexity of today's automation needs. Direct conversion of any petri net into ladder logic is achieved by adopting the petri net concept of using tokens as the main mechanism for controlling the flow of the sequential logic. Hence, each place within the petri net corresponds to a place within the program. Each action at a petri net place corresponds to an action at a ladder place, and each transition within the petri net involving a movement of tokens between ladder places. This simulated movement of tokens is achieved by deploying separate counters at each ladder places and the flow of petri net tokens is simulated by counting down and counting up the counters at appropriate ladder places within the program. Thus, each ladder place within the program has an associated counter, and the current count at the ladder place represents the number of tokens that would be at the corresponding place within the petri net.

5. DESIGN OF A DECS FOR THE MANUFACTURING SYSTEM

5.1. Sequential Control Petri Net for the Manufacturing System

The Sequential Control Petri Net (SCPN) for the Manufacturing System is shown in Fig. 4. The overall task can be broken down into four sub tasks of: conveyor operation; sorting; assembly chute and assembly process, as mentioned above.
The conveyor operation is controlled by two places; P1 and P2. A token in place P1 indicates the system is shut down. A token in place P2 indicates the system is operational.
Sorting is controlled by three places; Pso, P3, and P5. The transition Tr3 is enabled and a token is passed if the system is operational, there is a ring in the sort area, and there is space in the assembly chute. A token in the timed place P3 actuates the sort solenoid for 0.5 sec. If there is no token remaining in place P5, this indicates that assembly chute is full and other rings are inhibited for entering the chute.
The assembly chute is controlled by Place P4, which can have five rings. This is arranged by means of the number of the tokens in the limited capacity place P5.
Assembly process is controlled by 4 places; P6, P7, P8 and Psi. A token in the timed place P6 allows time for both components to travel down the assembly chute and for assemblies to clear the assembly area. A token in the timed place P7 actuates the assembly solenoid, which places a plastic ring at the assembly point. The Place P8 is used to show the assembly area and a token in this place shows the presence of a ring in the assembly area.

5.2. Ladder Logic Program for the Sequential Control Petri Net

The ladder logic program, shown in Fig. 5, is obtained for the Sequential Control Petri Net (SCPN) by using the TPLL method.

In order to convert the SCPN into TPLL a counter is deployed at each ladder place to represent the tokens at that place. The TPLL method is then applied to the SCPN shown in Fig. 4. The resulting ladder logic for a Siemens S5 PLC is shown in Fig. 5.
The system is initialized by rung 0. The conveyor operation is controlled by the rungs 1 and 2. Sorting is controlled by the rungs 3 and 4. The assembly chute is controlled by the rung5. The assembly process is controlled by the rungs; 6, 7 and 8. (Fig. 4).

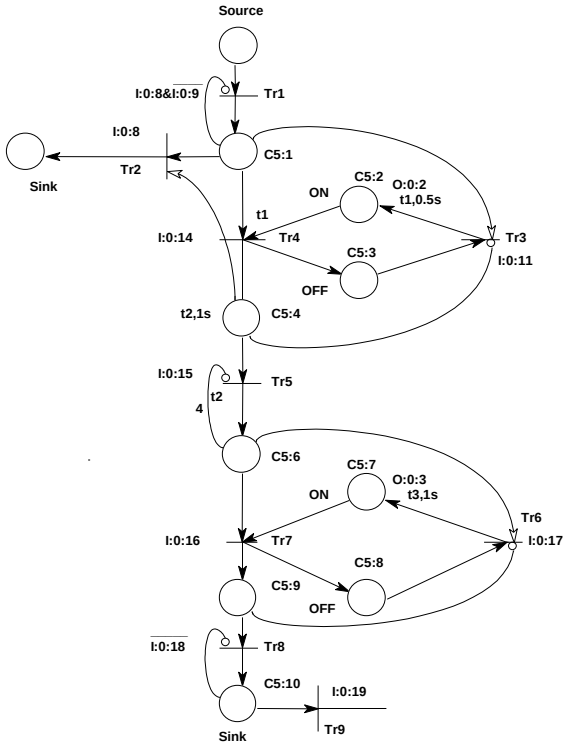
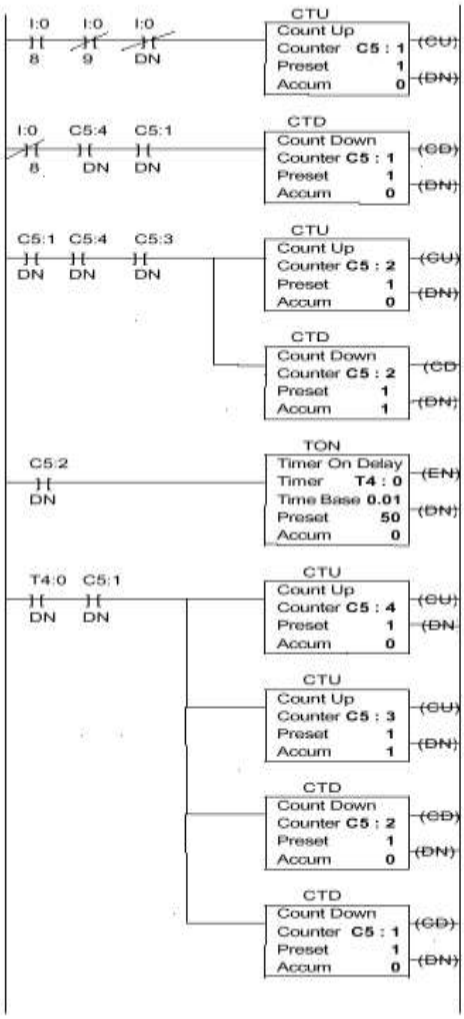
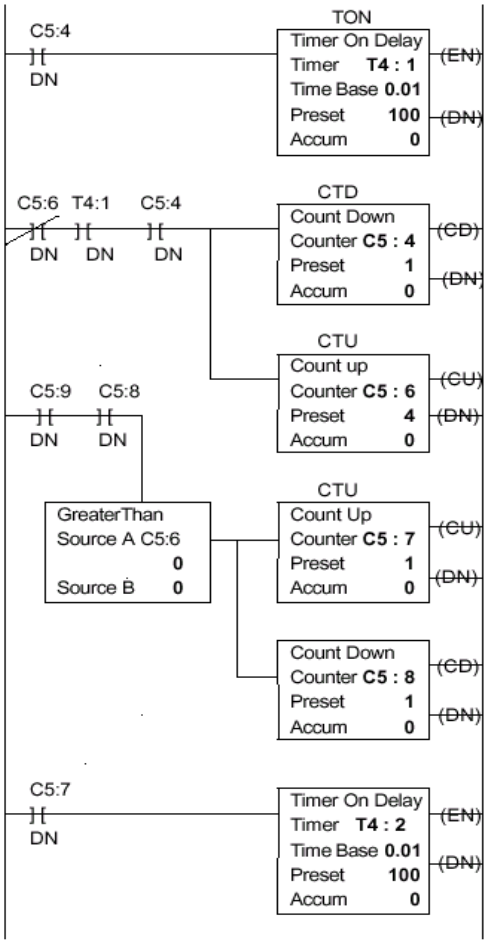


Figure 4. Sequential Control Petri net for the Manufacturing System

Finally, all the timer operations and actions are controlled by the rungs; 9, 10, 11 and 12**.**


I:0
8
I:0
9
I:0
DN
CTU
Count Up
Counter C5 : 1
Preset 1
Accum 0
(CU)
(DN)
I:0
8
C5:4
DN
C5:1
DN
CTD
Count Down
Counter C5 : 1
Preset 1
Accum 0
(CD)
(DN)
C5:1
DN
C5:4
DN
C5:3
DN
CTU
Count Up
Counter C5 : 2
Preset 1
Accum 0
(CU)
(DN)
CTD
Count Down
Counter C5 : 2
Preset 1
Accum 1
(CD)
(DN)
C5:2
DN
TON
Timer On Delay
Timer T4 : 0
Time Base 0.01
Preset 50
Accum 0
(EN)
(DN)
T4:0
DN
C5:1
DN
CTU
Count Up
Counter C5 : 4
Preset 1
Accum 0
(CU)
(DN)
CTU
Count Up
Counter C5 : 3
Preset 1
Accum 1
(CU)
(DN)
CTD
Count Down
Counter C5 : 2
Preset 1
Accum 0
(CD)
(DN)
CTD
Count Down
Counter C5 : 1
Preset 1
Accum 0
(CD)
(DN)


C5:4
DN
TON
Timer On Delay
Timer T4 : 1
Time Base 0.01
Preset 100
Accum 0
(EN)
(DN)
C5:6 T4:1 C5:4
DN DN DN
CTD
Count Down
Counter C5 : 4
Preset 1
Accum 0
(CD)
(DN)
CTU
Count up
Counter C5 : 6
Preset 4
Accum 0
(CU)
(DN)
C5:9 C5:8
DN DN
GreaterThan
Source A C5:6
0
Source B 0
CTU
Count Up
Counter C5 : 7
Preset 1
Accum 0
(CU)
(DN)
Count Down
Counter C5 : 8
Preset 1
Accum 0
(CD)
(DN)
C5:7
DN
Timer On Delay
Timer T4 : 2
Time Base 0.01
Preset 100
Accum 0
(EN)
(DN)

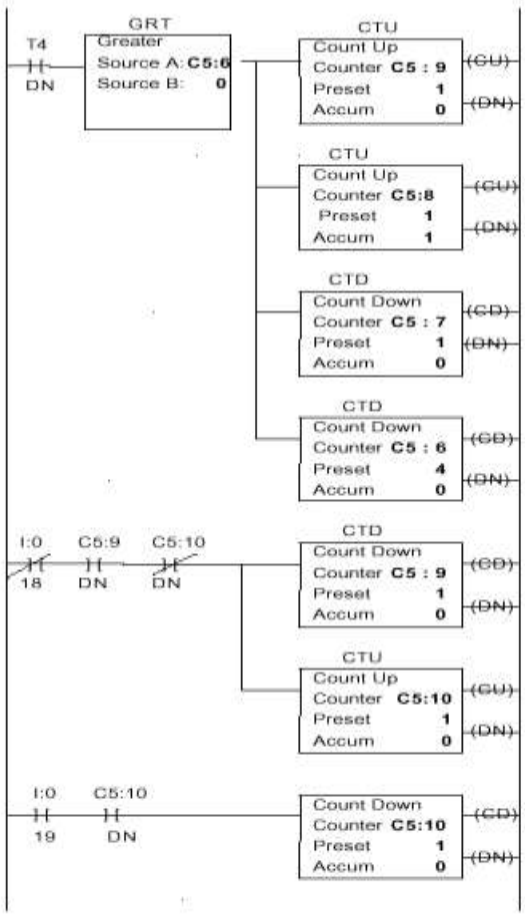


Figure 5. PLC Programming for Petrinet model of the manufacturing system in Fig. 4

6. MULTI-LEVEL LOGIC CONVERSION OF LLD

Two-level logic conversion is often implemented as multi-level petri net models of complex manufacturing systems. The fine granularity of multiple- level petri net models provides us with several degrees of freedom in conversion design of LLD that may be exploited in optimizing area and delay as well as satisfying specific constraints, such as different timing requirements 0 and n2-1. on different input/output paths. Thus multiple-level logic models are preferred to two-level logic implementations such as complex and dynamic manufacturing systems.
In multi-level logic control of manufacturing systems, there are three main input conditions for firing transitions, which define whether a transition is enabled to fire.

1. If the input place of a transition t0 is connected to the transition with a direct ordinary arc then transition t0 is said to be enabled when the input place p1

contains at least the number of tokens equal to the weight of directed ordinary arc connecting p2 to t0 unless when there are no p2 inhibitor arcs. The inhibitor arc which has 3 weighted arc doesn't allow fire t0 when p2 has 2 and below. This is shown in Fig. 6 with LLD program. When the number of tokens of p2 becomes less than 3 t0 is enabled and greater than or equal to 3 t0 is blocked. This is the LESS THAN or EQUAL instruction in the Allen Bradley PLC terminology. Therefore, in order to enable t0, p2's counter accumulator value has to be has number of tokens between

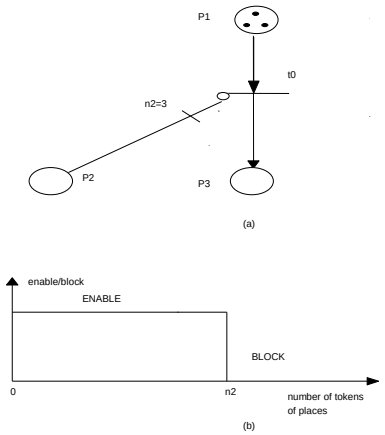


Figure 6. Illustration of enabling rules of less than or equal instruction

2. If the input place of a transition t0 is connected to the transition with a direct ordinary arc then transition t0 is said to be enabled when the input place p1 contains at least the number of tokens equal to the weight of directed ordinary arc connecting p2 to t0 unless when there is no p2. The enabling arc which has 3 weighted arc doesn't allow fire t0 when p2 has 3 and over. This is shown in Fig. 7 with LLD program. When the number of tokens of p2 becomes greater than or equal to three t0 is enabled and less than three t0 is blocked. This is the GREATER or EQUAL instruction in the Allen Bradley PLC terminology. Therefore, in order to enable t0, p2 has to be has between n2 and maximum accumulator value of the p2's counter.

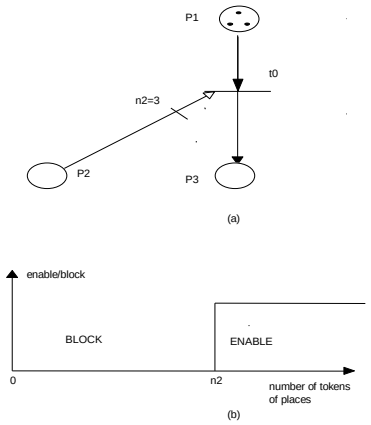


Figure 7. Illustration of enabling rules of GREATER OR EQUAL instruction

3. If the input place of a transition t0 is connected to the transition with a direct ordinary arc then transition t0 is said to be enabled when the input place p1 contains at least the number of tokens equal to the weight of directed ordinary arc connecting p2 to t0 unless when there are no p2 inhibitor and enabling arcs. The inhibitor arc and enabling arc, which have 6 and 3 weighted arcs, allow fire when p2 has 3 and over tokens and p2 has five and less to is enabled and at the other boundaries and ranges t0 is blocked. This is the LIMIT instruction in the Allen-Bradley PLC terminology. This is shown in Fig. 8 with LLD program. As it is special case, inhibitor weighted arc number equal to enabling weighted arc number, in this case, LLD instruction will be EQU instead of LIM instruction. Therefore, in order to enable t0, p2's counter accumulator has to be has between n2 and maximum p2's counter accumulator value, and p2's counter accumulator has to be has between 0 and n4-1 values.

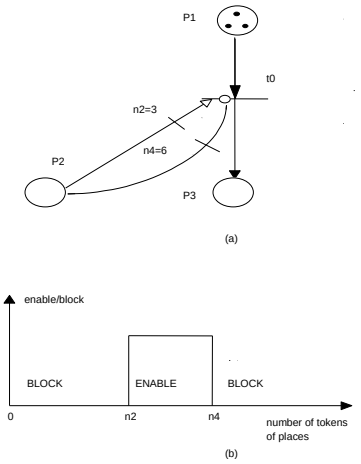


Figure 8. Ilustration of enabling rules of LIMIT instruction

7. CONCLUSION

As the requirement for more flexible manufacturing systems increases the need for a formal method for the design of Discrete Event Control Systems (DECS), which controls the manufacturing system, becomes more important. Petri nets represent the most effective method for the design of such DECS. The conversion of such Petri nets into ladder logic has recently been greatly simplified through the advent of the TPLL methodology. The new technique does not involve any intermediate steps and provides a straightforward but powerful method by which to implement ladder logic code. In this paper the fundamental Petri net structures and their associated TPLL equivalents have been described. Finally, design of a DECS for a manufacturing system has been considered: A Control Petri net has been developed for this manufacturing system and then the TPLL methodology has been used to derive structured ladder logic code.
This paper proposes a systematic approach for design supervisory controller for Discrete Event Systems and their Ladder Logic Diagrams. The method is based on Petri Nets, which is used to model the systems. It involves defining the control policy, simplifying the control policy using Espresso software (using Espresso Boolean logic reduction algorithm to re-define the control policy as set of steering places which enable controllable transitions and implementing the steering place control policy in the form of inhibitor and enabling arcs to the appropriate controllable transition) to form the compiled controller and then the method is used to drive structured Ladder Logic code. A design example is presented to clarify the full procedure. The resulting method known as EDICT can thus provide maximally permissive compiled supervisory solution to the forbidden state problem.

REFERENCES

[1] P.J.Ramadge and W.M.Wonham, “Supervisory Control of Discrete Event Process”, SIAM Journal on Control and Optimization, Vol.25, No.1, pp.206~230, Jan. 1987.

[2] P.J.Ramadge and W.M.Wonham, “Modular Feedback Logic for Discrete Event Systems”, SIAM Journal on Control and Optimization, Vol.25, No.5, pp1202~1218, Sept. 1987.

[3] A.Giua, “Petri Net Techniques for Supervisory Control of Discrete Event Systems”, Proc. of 1st Int. Workshop on Manufacturing and Petri Nets, Osaka Japan, pp.1~21, June 1996.

[4] E.Yamalidou, J.O.Moody and M.D.Lemmon, “Feedback Control of Petri Net Based on Place Invariants”, Automatica 32(1), pp.15~28,1996.

[5] R.K.Brayton, G.D. Hachtel, C.T. McMullen and A.L. Sangiovani-Vincentelli, “Logic Minimization Algorithms for VLSI Synthesis”, Kluwer Academic Publishers, 1984.

[6] A.H. Jones, Y.M. Li and S.B. Kenway, “Edict Supervisory Control of Discrete Event Systems”, UKACC International Conference on Control 2000, Cambridge, UK, 2000.